AMENDMENTS TO THE DRAWINGS

Please substitute the attached Replacement Sheet for Figure 3.

REMARKS

Status of the Claims

Claims 1, 5, 7, 8, 12, 14, 18, and 20 are currently present in the Application, and claims 1, 8, and 14 are independent claims. Claims 1, 5, 7, 8, 12, 14, 18, and 20 have been amended, and claims 2-4, 6, 9-11, 13, 15-17, and 19 have been canceled. Applicants are not conceding in this Application that those claims are not patentable over the art cited by the Examiner, as the present claim amendments and cancellations are only for facilitating expeditious prosecution of the present Application. Applicants respectfully reserve the right to pursue these and other claims in one or more continuations and/or divisional patent applications.

In particular, Applicants have amended independent claim 1 to include limitations previously found in dependent claims 2-4 and 6, and have therefore canceled claims 2-4 and 6. Similarly, Applicants have amended independent claims 8 and 14 to include limitations previously found in dependent claims 9-11 and 13, and 15-17 and 19, respectively, and have therefore canceled claims 9-11, 13, 15-17, and 19. Further support for Applicants' amendments is found, for example, in Applicants' specification on page 13, line 16 through page 16, line 10 (also see Figure 4).

Examiner Interview

Applicants wish to thank Examiner Zhe and Examiner Del Sole for the courtesy extended to Applicants' attorney during a telephone interview on June 19, 2007. During the interview, claim 1 was discussed with regard to the Silen reference (see below for further details). Applicants' attorney explained that Applicants teach and claim a thread executing a critical section of code. When a preemption event is detected during the critical code section, the current thread's priority is boosted by a priority offset amount. The boosted priority of the current thread is then compared to the priority of the preemption event, and if the boosted priority of the currently executing thread is higher than the priority of the preemption event, the currently executing thread is allowed to continue its execution. However, in contrast to Silen (and as discussed in further detail below), if the boosted priority of the currently executing thread is not higher than the

priority of the preemption event, the currently executing thread is preempted. Applicants' attorney suggested amending the independent claims to clarify this aspect of Applicants' invention. Examiners Zhe and Del Sole indicated that they would perform further searching after receiving Applicants' formal response. No agreement was reached regarding the claims during the interview.

Drawings

The Office Action does not indicate whether the formal drawings, filed with the Application on July 24, 2003, are accepted by the Examiner. Applicants respectfully request that the Examiner indicate whether the formal drawings are accepted in the next communication.

Applicants further note that a Replacement Sheet is hereby submitted for Figure 3 in order to correct an inadvertent, typographical error.

Claim Rejections - Alleged Anticipation Under 35 U.S.C. § 102 and Alleged Obviousness Under 35 U.S.C. § 103

Claims 1-5, 7-12, 14-18, and 20 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Silen, U.S. Patent No. 5,333,319 (hereinafter Silen). Applicants respectfully traverse the rejections under 35 U.S.C. § 102. Claims 6, 13, and 19 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Silen in view of Janssen et al., U.S. Patent Publication No. 2006/0037025 (hereinafter Janssen). Applicants respectfully traverse the rejections under 35 U.S.C. § 103.

As noted above, Applicants have amended independent claim 1 to include limitations previously found in dependent claims 2-4 and 6, and have therefore canceled claims 2-4 and 6. Similarly, Applicants have amended independent claims 8 and 14 to include limitations previously found in dependent claims 9-11 and 13, and 15-17 and 19, respectively, and have therefore canceled claims 9-11, 13, 15-17, and 19. Applicants respectfully submit that none of the cited art, either alone or in combination, teaches or suggests amended, independent claims 1, 8, and 14. Using independent claim 1 as an exemplary claim, Applicants claim the following:

- indicating that the execution thread needs a higher priority by updating a user mode accessible data area, the indicating performed without increasing a priority corresponding to the execution thread and the indicating including setting a critical section flag within the user mode accessible data indicating that the execution thread is entering a critical section of code;
- updating a priority offset amount prior to the execution thread entering the critical section of code, whereupon the priority offset amount is included in the user mode accessible data;
- during execution of the critical section of code:
    - detecting a preemption event, wherein the preemption event has a preemption event priority;
    - reading the user mode accessible data area in response to the detected preemption event;
    - determining whether a priority applied flag and the critical section flag have been set;
    - in response to detecting that the critical section flag has been set and the priority applied flag has not been set, raising the execution thread's priority by the priority offset amount and setting the priority applied flag;
    - after raising the execution thread's priority:
        - in response to the preemption event priority being greater than the execution thread's priority, preempting the execution thread; and
        - in response to the preemption event priority not being greater than the execution thread's priority, allowing the execution thread to continue executing.

Silen purports to teach dispatching a work unit at a user address space priority, rather than a resource manager address space priority (Silen, col. 4, lines 30-35). Silen further purports to teach allowing an interrupted task to continue to completion rather

than preempt the interrupted task and make a task switch (Silen, col. 6, lines 47-50). Janssen purports to teach lowering the priority levels of one or more threads if their measured relative use exceeds a certain escalation threshold for the duration of an escalation time period (Janssen, Abstract).

However, neither Silen nor Janssen teaches or suggests several key elements of Applicants' independent claims. Applicants teach and claim "setting a critical section flag" to indicate that an "execution thread is entering a critical section of code." During the execution of the critical section of code, a preemption event is detected, "wherein the preemption event has a preemption event priority." A determination is made regarding whether "a priority applied flag and the critical section flag have been set." As shown in Applicants' specification with regard to Figure 4, the priority applied flag is set to indicate that the execution thread's priority has already been boosted. Therefore, if the critical section flag is set and the priority applied flag is not set, this means that the execution thread is executing a critical section of code but has not yet had its priority boosted.

Applicants further teach and claim "in response to detecting that the critical section flag has been set and the priority applied flag has not been set, raising the execution thread's priority by the priority offset amount and setting the priority applied flag." After the execution thread's priority has been raised, Applicants teach and claim that the execution thread may or may not be preempted. If the preemption event priority is greater than the execution thread's priority, the execution thread will be preempted. On the other hand, if the preemption event priority is not greater than the execution thread's priority, the execution thread will continue executing. This is in direct contrast to Silen. Silen discloses an EPRTY (escalated dispatching priority flag) that "when set, overrides current dispatching affinity and forces the assigned affinity to the MVS master address space" (Silen, col. 5, lines 41-44). Silen discloses that when a work unit is made available for dispatch, its ready work element is placed in a ready work queue. The ready work queue to which the ready work element is assigned depends on the setting of the EPRTY flag. If the EPRTY flag is non-zero, the ready work element establishes temporary priority dispatching affinity with the MVS master address space

(Silen, col. 6, lines 29-55). In other words, if the EPRTY flag for a ready work element is non-zero, the particular ready work element is always assigned the highest priority in the system, i.e. the MVS master address space priority. This means that the particular ready work element will not be preempted. As stated in Silen, "[t]his can result in work units experiencing a reduction in preemption during such critical code sequences, by delaying a task switch until the effected code sequence has completed and priority dispatching associated with current dispatching affinity is reenabled" (Silen, col. 6, lines 50-55).

Silen gives a further example in its Figure 6. A lock manager alters the priority of a lock owner so that the lock owner's work unit is "temporarily handled at the same priority level as the high priority waiter for the duration the lock is held by that owner" (Silen, col. 9, lines 56-64). In this example, if the dispatching priority of the lock requestor is higher than the dispatching priority of the lock owner, the priority of the lock owner is set equal to the priority of the lock requestor, such that the current lock owner is not preempted by the requestor.

Silen appears to be discussing two different implementations of boosting the priority of a work unit. In one example, a ready work element has an EPRTY flag that is non-zero and therefore the work unit is given special priority, i.e. the priority of the MVS master address space (Silen, col. 5, line 8 through col. 6, line 55). In the other example, the lock manager raises the priority of a current lock owner so that it is equal to the priority of a lock requestor (Silen, col. 9, line 56 through col. 10, line 32). However, in both of these examples, the current work unit is NOT preempted. Silen does not teach or suggest raising a work unit's priority and then re-checking to determine if the preemption event (such as a lock requestor) has a higher priority. This is in direct contrast to Applicants' claimed invention, where the currently executing thread may or may not continue executing, even after it has had its priority boosted. Applicants specifically teach and claim "after raising the execution thread's priority," that there are two possible outcomes, i.e. "in response to the preemption event priority being greater than the execution thread's priority, preempting the execution thread, and "in response

to the preemption event priority not being greater than the execution thread's priority, allowing the execution thread to continue executing."

Silen always appears to allow the currently executing work unit (or lock owner) to continue executing. Because the currently executing work unit is given the highest priority (or alternately, a priority equal to the priority of the higher-priority lock requestor), Silen discloses that the current work unit (or lock owner) always receives enough of a priority boost to continue executing without being preempted. This teaches away from Applicants' claimed invention, where the current execution thread has its priority boosted, but may or may be allowed to continue executing. Applicants specifically teach and claim that the priority boost may not be enough of a boost to allow the current execution thread to avoid being preempted.

Janssen discloses temporarily boosting a thread's priority by adding a factor to the thread's basic priority level or by adding a time-varying amount to the thread's basic priority level (Janssen, paragraph [0035]). Janssen teaches lowering a thread's basic priority level if the thread's relative use exceeds a certain escalation threshold during an escalation time period of a certain length. This prevents one thread, or one process including many threads, from dominating system time and resources (Janssen, paragraphs [0038] through [0041]). However, Janssen does not teach or suggest raising an execution thread's priority by a priority offset amount, as taught and claimed by Applicants. Janssen further does not teach or suggest that after the execution thread's priority has been raised, the execution thread may or may not be preempted, based on whether or not the preemption event priority is greater than the execution thread's priority, as taught and claimed by Applicants.

As discussed above, neither Silen nor Janssen nor a combination of the two teach or suggest Applicants' claimed invention. Further, Silen actually teaches away from Applicants' claimed invention because Silen raises the priority of a current work unit (or lock owner) such that it can not be preempted, whereas the priority boost taught and claimed by Applicants may or may not be sufficient to prevent an execution thread from being preempted. For the reasons set forth above, Applicants respectfully submit that independent claims 1, 8, and 14 are not anticipated by, and are patentable over,

the cited art. Therefore, Applicants respectfully request that independent claims 1, 8, and 14, and the claims which depend from them, be allowed.

Conclusion

As a result of the foregoing, it is asserted by Applicants that the remaining claims in the Application are in condition for allowance, and Applicants respectfully request an early allowance of such claims.

Applicants respectfully request that the Examiner contact the Applicants' attorney listed below if the Examiner believes that such a discussion would be helpful in resolving any remaining questions or issues related to this Application.

Respectfully submitted,

By /Leslie A. Van Leeuwen, Reg. No. 42,196/
Leslie A. Van Leeuwen, Reg. No. 42,196
Van Leeuwen & Van Leeuwen
Attorneys for Applicant
Telephone: (512) 301-6738
Facsimile: (512) 301-6742